`FTPMAN` Snapshots—miscellaneous notes

Obsolete notions

Tue, Mar 25, 1997

Digitizer Sharing

In order to preserve the desirable feature that many users can share access to the swift digitizer data, `FTPMAN` support may allow for looking without touching. The `FTPMAN` request protocol includes parameters such as digitize rate, clock event selection, delay, and number of points. The status reply message that updates the client on the progress of collecting the snapshot data includes provision for the *server* to declare these parameters. By arranging for the server to examine the current programmable settings—analogous to the adjustments of a camera—and reply with the current values of these parameters, we can achieve the goal of allowing many users to access the same waveform at the same time without interference. Of course, since the hardware parameters are settable on this board, a user can change them, if needed, through a parameter page. But the user who does this knows that such changes are global; *i.e.*, they affect all users of the waveforms. Note that the hardware design implies that any such changes are common to all 8 channels of a given swift digitizer board, not only those of current user interest.

A caveat is in order. The status reply to a snapshot request allows the server to "correct" several parameters, as described above, but this set of parameters is common to all signals in the request. Thus, one cannot ask to collect two signal waveforms in one request unless they share the same parameter set. It may be likely that these parameters are configured in hardware once and seldom changed. *(This sharing consideration is ultimately rejected in favor of giving the user more flexible control over the snapshot parameters. This topic is revisited later.)*

We could allow a client to specify the parameters, but arrange that the server restores them after the snapshot has been taken. Because there may be more than one user of `FTPMAN` for any of the 8 signals, we might check, upon receiving new data, whether the parameters used to make the snapshot are the same as expected. A queue can be used that a local application reads to specify parameters for snapshots to be taken. The application could then capture the data and return a pointer to a memory block containing such data in response. Using such queuing may make it difficult to provide correlated waveforms across different nodes.

Swift support without `FTPMAN`

Since the waveforms are deposited by the hardware into fixed areas of memory, one can easily plot the waveform that exists there, even allowing for overlap, if desired; whatever waveform is found there is the last snapshot taken. This very informal access to the swift digitizer waveforms works in many cases. But if `FTPMAN` is also running, one may get interference from `FTPMAN`-initiated snapshots, if support is allowed for varying the snapshot parameters as discussed above. Also, when a network request is satisfied, the digitizer may be building a new waveform, so that the data is part new, part old. We need to avoid such occurrences.

Access to swift digitizer waveforms could be provided via ordinary network requests, rather than low-level access via memory address. A new listype could support such access. Its ident might be merely a channel#. The system needs to find out whether that channel supports a swift digitizer waveform. The returned data can include a header that indicates the snapshot parameters. If the parameters change, the client could detect that. For an overlapped plot, the client might refuse to plot unless the parameters return to the same ones that were in effect the first time data was returned. For the single-trace mode, each plot could be properly labeled with the conditions used.

SWFT local application—first try

Four channel#s are parameters to this local application. The readings of these parameters sequence the operation of the swift digitizer board. Install selected parameter values into the registers, based upon the channel readings. Establish and enable the delay timer interrupt. Arm the board. Each cycle, monitor a counter that is incremented by the occurrence of the interrupt. If the counter changes, the delay timer has finished and the digitizer has started. Each 15 Hz cycle, check the value of the memory address register. When it becomes reaches (#points*8), digitization is complete; all eight waveforms have been collected in the 64K on-board memory.

Before repeating the measurement, we need to allow for a requester to capture the memory as desired. But how can we do this? FTPMAN is one possible requester, but we need to alert FTPMAN that the waveform is ready. Perhaps a state variable can inform FTPMAN of this progress. A non-FTPMAN user is any (probably a Classic protocol) requester using a new listype designed for access to this waveform data. How can we be sure that such a user can capture the data from the hardware memory before re-arming the hardware to collect another waveform?

In the FTPMAN case, we must try to maintain synchronous operation across multiple nodes so that server-based access works. Suppose that FTPMAN captures the delay timer interrupt counter value to signal that it has detected any data it needs from the hardware memory. This captured value would be stored in the related capture memory block for the given request. Perhaps SWFT could find such requests to see whether the counter has been updated to match the present counter value. If it matches, then FTPMAN has no further need of keeping the waveforms frozen in the hardware memory.

In the case of the use of a special listype for the purpose, such as from the Macintosh Parameter Page program, SWFT can check whether such requests are active. If so, it could notice whether they have been updated with the latest delay time interrupt counter value.

Re-visit changing parameters question

The swift digitizer hardware can run in systems that have FTPMAN support as well as ones that do not. If FTPMAN is active, suppose we allow a request to establish the programmable parameters in the hardware. A second user may do the same thing. If the parameters are different, it will mean that the "first hog to the trough" loses, assuming that both users are active simultaneously. It is expected that most times, only one user will be actively interested in such data. At least this provision would enable that user to feel in control of the digitizing parameters.

A non-FTPMAN user would operate the three channel parameters as needed. This could interfere with an active request from another user, though. *Access to swift digitizer data without FTPMAN needs more thought.*

Access to waveforms via new listype

There may be two new listypes supported, each using a channel# ident. One can return a short data structure about the conditions of the measurement of the waveform currently existing in the hardware buffer. This could include the four high-level parameters described above that are used as input to SWFT. It could also include an 8-byte date and time structure. The interrupt counter value can also be included. The requester can label a plotted waveform with this information. The second listype provides for access to the waveform data itself. By watching the data in such a request, one can know when it has been updated with the latest info. Such a user would have to request the waveform itself upon learning that a new one is available. One cycle after that, a new waveform can be collected, allowing one cycle for the requester to send a one-shot request for that waveform.

This scheme assumes that the requester can access the waveform data readily. If he has to collect 4 signals of 4K points each (32K bytes), this may be a tall order! The point is that we do not want to copy waveform data out of the hardware buffer except when necessary. It is necessary for FTPMAN because of its protocol design. We do not want to stall the collection of new waveforms for possible other clients just because an FTPMAN user is not finished looking at the display of a waveform just captured.

For the non-FTPMAN case, the request might include the parameters, or the parameters could be adjusted separately, to be used the next time a measurement is called for. The data can be returned at whatever rate was requested. The #bytes requested should be enough for the data from at least one signal. If this is not possible, then the reply data will have to be delivered in parts. When all has been returned, the status can indicate this, and another measurement can be permitted, if there is something waiting in the message queue.

```
Request data ident
        node
        chan

Setting data ident
        node
        chan

Setting data
        event (2)
        delay (2)
        rate (2)
        #pts (2)

Reply data
        status (2)
        event (2)
        delay (2)
        rate (2)
        date/time (8)
        #pts (2)
        data index (2)
        data words (n*2)
```

Suppose, in an effort to prevent interference between multiple users, we only try to satisfy one user at a time. The local application SWFT reads snapshot parameters from a message queue, installs the parameters into the swift digitizer registers, then makes the measurement. When finished it does nothing further until there was confirmation of no further need for the data. In this case, copying would not be necessary. For FTPMAN, though, copying could be useful, because the user may keep the request active for a long time.

For Classic protocol access to this data, copying of waveform data will not be done, in order to prevent unnecessary time used for that purpose. (The reason for adding a memory to the swift digitizer board was to avoid having to do this.) The hardware buffer must therefore remain stable while the data is emptied out to a requester. A requester makes a periodic request for such data. Status, parameters, and data (when available) will be included in the replies. The status will indicate complete when the delivery of all data is finished. The reason for delivering the data in pieces is that up to 4096 words (8192 bytes) of data are available *for each channel* in the request, so it cannot all fit into a single datagram. The request period is limited to 15Hz, so a user should choose the buffer size with care so that the hardware buffer doesn't hold up a new measurement inordinately long. *Again, non-FTPMAN use of the swift digitizers needs further thought.*

Auto-repeat of last measurement

If the queue is empty when SWFT is ready to make another measurement, one might enable a repeat of the previous parameters used. This would keep the memories fairly current. If an infrequent event were chosen, one might want to have a way of canceling the current parameter set. To this end, one might like to be able to view the contents of the queue. The contents of the queue might be 16 bytes altogether, assuming 2 bytes for the delay in μs, 8 bytes for date/time.

```
evnt  delay,ms  rate,kHz  #points
0014   16.02     100       4096
```

Update of replies to waveform listype

If status last sent was incomplete, continue to return updated status only. But watch for completion status. When this occurs, return such status, and commence returning data as will fit in buffer given.

Who would place a new entry into the queue? At first, FTPMAN would do it. Next a new setting routine would do it, in order to support non-FTPMAN (Classic) users. This means that the time included in the queue entry must be the time-of-day that the entry was added to the queue, not the time-of-day the measurement was made.